Agilné metódy vývoja softvéru a rozsah projektu

MARTIN KOMARA

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
martin.komara@gmail.com

Abstrakt. Vplyv softvérových systémov na náš život je obrovský a rastie každým dňom. Softvér sa stal nevyhnutným pre fungovanie ľudskej spoločnosti. Softvérové systémy patria k najzložitejším výtvorom ľudstva vôbec, čo spôsobuje problémy pri ich tvorbe a prevádzke. Vysoká zložitosť softvérových systémov viedla ku vzniku veľkého množstva prístupov a metód, ktoré sa snažili o systematický prístup k tvorbe softvéru. Napriek tomu len málo systémov bolo dodaných do prevádzky za vopred dohodnutých podmienok. V situácii, keď sa zvyšuje tlak, ktorý núti vytvárať kvalitný softvér za menej peňazí a za kratší čas, vznikli nové, tzv. agilné metódy. Agilné metódy vznikli zo skorších prístupov k rýchlemu vývoju aplikácií (RAD), ktoré rozpoznali, že spomedzi obmedzení projektu tvorených časom, cenou a rozsahom má práve rozsah najvyššiu mieru neurčitosti. Preto sa agilné metódy, na rozdiel od tých tradičných, nepokúšajú určiť rozsah projektu v úvodných fázach, ale pracujú s používateľskými požiadavkami oveľa flexibilnejšie. Tento príspevok rozoberá nové zaujímavé postupy na určovanie rozsahu projektu pri použití agilných metód vývoja softvéru.

Úvod

Odkedy bol vymyslený tento svet, stretávajú sa jedinci ľudského rodu s najrôznorodejšími obmedzeniami. Najskôr išlo o rieku, ktorá zabraňovala prapredkom dostať sa k zdrojom tukov, bielkovín a cukrov, pokojne sa pasúcim na druhej strane. Potom to bol oceán, ktorý sa stal poslednou zastávkou mnohých dobrodruhov (traduje sa napríklad príbeh istého Janovčana, ktorý neumrel na mori od hladu a smädu len vďaka tomu, že sa medzi Európou a Áziou, kam sa dobrodruh pôvodne chcel dostať, zhodou okolností vyskytol ešte jeden dovtedy neznámy kontinent). Neskôr ľudstvo vypliešťalo oči, keď zistilo, že sa aj tak nedostane nikam, pretože sa nachádza na jednej veľkej guli. A aby sme náhodou neboli nezdravo optimistickí po objave prvej kozmickej rýchlosti, veľmi rýchlo bolo ľudstvo oboznámené s drobným obmedzením nazývaným hraničná rýchlosť vo vesmíre. Existujú jedinci, ktorí sa rozhodli bojovať s obmedzeniami, ktoré život prináša. Títo sa

Manažment v softvérovom inžinierstve, máj 2005, s. 1-6.

dajú rozpoznať podľa toho, že ich hroby bývajú vytvorené ad-hoc, navŕšené z dostupných materiálov. My ostatní zbabelí, túžiaci po dôstojnom mieste posledného odpočinku, sa musíme naučiť s týmito obmedzeniami žiť.

Taký softvérový projekt, napríklad. Na softvérový projekt sa vzťahujú najmä tri obmedzenia: Čas, cena a rozsah.

Diabolská trojica

Pre mnohé projekty je čas absolútne rozhodujúcim kritériom. Je to najmä v prípadoch, keď je tento dátum daný externými okolnosťami, na ktoré nemá organizácia nijaký vplyv. Príkladom by mohla byť banka, ktorá musí upraviť svoj informačný systém tak, aby spĺňal nové legislatívne požiadavky pred tým, než legislatívna úprava nadobudne účinnosť. Iným príkladom takéhoto projektu môže byť tvorba eseje, ktorú musí študent odovzdať predtým, ako prof. Bieliková zruší možnosť vkladania do redakčného systému.

Cena je pre projekt mimoriadne dôležitá. Ako inak, veď ide o peniaze. Výsledná cena pozostáva z dvoch častí: nákladov a zisku. Náklady sú tvorené nákladmi na softvérové a hardvérové vybavenie (vrátane údržby), nákladmi na školenia, nákladmi na pracovnú silu a nákladmi na réžiu. Zatiaľ čo náklady na softvér a hardvér sa dajú dobre odhadnúť, náklady na pracovnú silu závisia od času, koľko bude daný projekt napokon trvať. Ako teda vidno, cena závisí od času.

Funkcionalita systému je tretie obmedzenie. Neschopnosť dodať prisľúbenú funkcionalitu v dohodnutom rozsahu znamená nedodržaný prísľub, čo sa prejavuje na obchodných vzťahoch a má nepriaznivé dopady na ďalšiu spoluprácu. Väčší rozsah projektu si samozrejme vyžaduje viac času a/alebo zdrojov. Z toho vyplýva, že čas a cena projektu sa odvíja od jeho rozsahu.

Tradičné metódy, tradičné problémy

Tradičný model manažmentu softvérového projektu sa zameriaval na zafixovanie rozsahu, z ktorého sa potom vychádzalo pri dohode so zákazníkom o cene a čase. Je tu však malý háčik. Ak je najskôr definovaný rozsah, vývojári sú nútení spraviť odhad úsilia, ktoré bude potrebné vykonať na uvedenie daného produktu do života. Problém, ktorý iste nepoteší, je, že spomedzi času, ceny a rozsahu má práve rozsah najväčšiu mieru neistoty. Neurčitosť vykonaných odhadov na základe takto určeného rozsahu je veľmi vysoká a môže vyústiť do situácie, že skutočné a odhadnuté množstvo potrebných zdrojov a času sa líši až o 200%.

V takejto situácii vznikol veľký tlak na manažérov, aby vytvárali presnejšie odhady. Vznikli nové techniky, ktorých využitie malo vyústiť do lepších, presnejších odhadov. Využívanie týchto techník však znamenalo stále väčšie a väčšie množstvo papierovania, s čím súvisela potreba veľkého množstva času na vytváranie, čítanie, konzultáciu a schvaľovanie. O škodách na lesoch ani nehovoriac. A to bol v skratke príbeh, ako sa zrodili tzv. ťažkotonážne, tradičné metódy vývoja softvéru.

Tradičné metódy vývoja softvéru definujú rozsah produktu pomocou rozpisu práce, počnúc vysokoúrovňovými požiadavkami, ktoré boli ďalej dekomponované na špecifickejšie požiadavky [3]. Tento postup dáva manažérovi k dispozícii presnejší odhad času a potrebných zdrojov (inými slovami výslednej ceny) celého projektu, pretože čím presnejšie je špecifikovaný projekt a potrebné činnosti, tým sú jednotlivé odhady lepšie. Akonáhle je vytvorený takýto rozpis prác, úlohou manažmentu rozsahu projektu ostáva sledovanie, aby nedošlo k zmene rozsahu, najmä formou nekontrolovanej zmeny požiadaviek.

Tieto metódy fungujú dobre, pokiaľ nie je produkt príliš zložitý a požiadavky sú konzistentné a zlučiteľné. Bohužiaľ, v praxi bývajú produkty príliš zložité na to, aby ich bolo možné úplne definovať, čo vyúsťuje do skutočnosti, že výstupy z týchto metód bývajú nespoľahlivé a zavádzajúce.

Dôkaz, že pod tlakom sa pracuje mimoriadne ťažko, zverejnila v roku 1994 The Standish Group, keď uverejnila štúdiu, podľa ktorej iba 16,2% softvérových projektov bolo dodaných načas, so špecifikovanou funkcionalitou, a nebol prekročený plánovaný rozpočet. Pri 31,3% projektov nebol dodržaný dohodnutý čas, cena alebo rozsah. Zvyšných 52,7% projektov bolo zrušených. Ako najčastejším dôvodom, prečo došlo k zrušeniu projektu, uviedli respondenti nekompletnú špecifikáciu požiadaviek [4].

V takejto situácii sa akiste nemožno čudovať snahám nahradiť tieto tradičné metódy novými, efektívnejšími metódami vývoja softvéru, ktoré by umožnili vyhnúť sa nutnosti dopredu vytvárať odhady, ktoré sa napokon aj tak ukážu ako nepresné.

Agilné metódy na scénu

Agilný znamená schopný pohybovať sa rýchlo, flexibilne, s vysokým stupňom zručnosti a kontroly. Agilné metódy sú navrhnuté tak, aby boli flexibilnejšie. Zatiaľ čo pri tradičných metódach prevládalo presvedčenie, že pri dostatočnom úsilí sa nám podarí vytvoriť kompetnú sadu požiadaviek na finálny systém, ktoré sa už v ďalších fázach životného cyklu nebudú meniť, agilné metódy pokladajú zmenu požiadaviek za niečo, čo nastane bez ohľadu na úsilie, ktoré vynaložíme na to, aby sme tejto zmene zabránili.

Za kľúčovú sa považuje komunikácia v tíme, ktorú nedokážu nahradiť ani moderné komunikačné prostriedky, ako sú napr. email, videokonferencie či mobilné telefóny. Kontakt zoči-voči stále zostáva najpriamejšou a najefektívnejšou formou komunikácie.

Ako už bolo uvedené, jednou z najčastejších príčin zlyhania softvérového projektu je nedostatočné zapojenie zákazníka do procesu vývoja. Zákazník, ktorý je k dispozícii vývojovému tímu alebo je priamo jeho súčasťou, je kľúčovou požiadavkou všetkých agilných metód.

Zákazník by mal vedieť správne zodpovedať všetky otázky vývojového tímu, mať právo robiť záväzné rozhodnutia a rozhodovať sa správne a na rozdiel od tradičných metód vývoja softvéru by mal byť k dispozícii nielen v úvodnej fáze projektu, ale počas všetkých fáz životného cyklu.

Vzhľadom na to, že zapojenie zákazníka je hlavným cieľom agilných metód vývoja softvéru, najčastejšia technika získania požiadaviek na systém je rozhovor, ktorý je zdrojom priamych znalostí o probléme. Je známe, že reťazové predávanie si informácií vedie k nedorozumeniam. Preto všetky agilné metódy zdôrazňujú priamu komunikáciu so zákazníkom. V prípade, že vznikne nejasnosť alebo sú požiadavky definované príliš povrchne, člen vývojového tému by mal kontaktovať priamo zodpovednú osobu a vyhnúť sa komunikácii cez tretiu osobu.

Dôležitou vlastnosťou agilných metód je, že zmena sa, na rozdiel od tradičných metód, považuje za prirodzený jav v rýchlo sa meniacom prostredí a nepokladá za potrebné vyvíjať úsilie na jej elimináciu. Agilné metódy sa zameriavajú na vytvorenie takého návrhu, aby bolo možné zmeny jednoducho realizovať [2].

Rozsah

Ako sa teda určuje rozsah pri použití agilných metód? Treba povedať, že určovanie rozsahu funguje diametrálne odlišne ako pri tradičných metódach. V prvom rade sa výsledný čas projektu neodvíja od rozsahu, ale práve naopak. Urobí sa toľko, na koľko je dostatok času, pričom zákazník má plnú kontrolu nad tým, ktorá časť sa bude implementovať.

Agilné metódy sú charakteristické iteratívnym vývojom. Plán, čo sa bude v danej iterácii vykonávať, sa určí na jej začiatku v spolupráci so zákazníkom. Dĺžka trvania iterácie je obyčajne krátka, od dvoch do šiestich týždňov. Počas každej iterácie dôjde k implementácie niekoľkých cieľov.

Požiadavky by mali smerovať k výsledkom, ktoré sú pre koncového zákazníka nejakým spôsobom užitočné. Požiadavky, ktoré sa majú implementovať, sú uložené v rade s prioritou. Túto prioritu určuje zákazník v spolupráci s vývojovým tímom. Bežnou praxou je najprv implementovať požiadavky s najvyššou prioritou, ktoré prinášajú najvyššiu hodnotu pre zákazníka. Počas vývoja sa zlepšuje porozumenie problému a nové požiadavky na systém sú pridávané. Priorizácia je opakovaná často počas celého procesu vývoja, aby sa zabezpečilo, že všetky požiadavky sú aktuálne.

Je potrebné si uvedomiť, že priorita jednotlivých požiadaviek sa mení v čase. Napríklad priorita požiadavky, ktorá smeruje k splneniu legislatívnej normy, sa bude zvyšovať s približujúcim sa termínom účinnosti danej normy.

Preto je dôležité získať prioritu pre každú požiadavku z rozsahu meniacu sa v čase. Manažér by sa mal dohodnúť so zákazníkom na rozsahu, ktorý bude obsahovať niekoľko požiadaviek s nižšou prioritou, ktorá sa bude postupom času zvyšovať. Vývojový tím zahrnie tieto požiadavky do plánu, avšak zákazník súhlasí s tým, že niektoré z nich sa nepodarí dokončiť načas v prípade neočakávaných problémov.

Ako sa zlepšuje pochopenie problému počas jeho vývoja, môže sa stať, že sa zmení priorita niektorých požiadaviek, alebo dôjde k jej úplnému vypusteniu, nakoľko sa zistí, že splniť danú požiadavku nie je potrebné pre zabezpečenie obchodných cieľov.

Dôležitou vlastnosťou v súvislosti s radom požiadaviek je možnosť dohodnúť sa na rezerve pre danú iteráciu. Najprv by malo byť odsúhlasené celkové množstvo požiadaviek, ktoré sa budú implementovať v danej iterácii, na základe odhadu potrebného úsilia a dátumu dodania. Výsledkom je celkový počet požiadaviek, ktoré sa budú implementovať v danej iterácii. Potom sa dohodne rezerva, čo znamená, že požiadavky sa rozdelia do troch kategórii: na tie, ktoré je absolútne nevyhnutné implementovať, tie, ktoré by bolo potrebné implementovať a tie, ktoré by bolo dobré implementovať.

Snaha o kategorizovanie požiadaviek do týchto kategórií však naráža na neochotu zákazníka označiť akúkoľvek požiadavku pre danú iteráciu za inú ako nevyhnutne implementovanú. Zákazník bude predpokladať, že takto kategorizované požiadavky nebudú nikdy realizované. Tento predpoklad je založený na zlej skúsenosti so spoločnosťami, ktoré nezrealizujú čo prisľúbia, a celkom určite nikdy nezrealizujú, čo neprisľúbia. V takomto prípade sa nedá dohodnúť na rozsahu projektu, čo znamená riziko pre celý projekt [1].

Záver

Agilné metódy vývoja sú výzvou pre softvérových manažérov a klientov, ktorí sa musia vzdať závislosti na objemnej dokumentácii a medzivýsledkoch, tvorených výstupmi z jednotlivých fáz životného cyklu programu. Nástroje, ktoré sa používajú namiesto objemnej dokumentácie a rozpisu prác sú najmä zlepšená komunikácia so zákazníkom, priorizácia požiadaviek a krátke trvanie iterácie, na ktorej konci je fungujúci softvér.

Agilné metódy opúšťajú snahu zadefinovať rozsah vytvorením kompletného zoznamu požiadaviek na začiatku projektu, nakoľko odhad celkového úsilia potrebného na dokončenie projektu na základe rozpisu prác má v sebe veľkú mieru neurčitosti.

Agilné metódy umožňujú prispôsobovať rozsah projektu meniacim sa podmienkam prostredia a zohľadňujú vývoj toho, čo zákazník požaduje na základe dojmov z práce so softvérom dodaným v predchádzajúcich iteráciách.

Používateľ má možnosť v spolupráci s vývojovým tímom zmeniť svoje požiadavky na začiatku každej iterácie, ako aj zmeniť prioritu už existujúcich požiadaviek tak, aby dokázal čo najlepšie naplniť svoje obchodné ciele. Agilné metódy týmto vnášajú do rozsahu projektu flexibilitu, aká nebola možná pri použití tradičných metód tvorby softvéru.

Použitá literatúra

1. Anderson, D.J., Schragenheim, E.: *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Prentice Hall, PTR, New York, 2003.

2. Beck, K., Andres, C.: *Extreme Programming Explained: Embrace Change, Second Edition*. Addison Wesley Professional, 2004.
3. Bieliková, M.: *Manažment v softvérovom inžinierstve*. 1999.
4. *Chaos Report, Few IS Projects Come in on Time, on Budget*. Computer World 12, 1995.

Annotation

Agile software development methods and project scope

Impact of software systems on everyday life is ample, and continues to grow day by day. Software became inevitable for the existence of human society. Software systems are accounted to be among the most complex creations of mankind at all, causing troubles with their creating and maintaining. High degree of complexity led to the creation of great number of approaches and methods for systematic approach to software system manufacturing. Despite this effort, very few systems have been delivered to customer on time, on budget, and within scope. In a situation where the pressure to create quality software with thinner budget and shorter time to market is constantly growing, new so called agile methods have emerged. They are based on earlier rapid application development (RAD) approaches, which recognised that among project constrains represented by time, cost and scope is the scope one single constraint with the greatest measure of uncertainty. It is for this reason why agile methods unlike traditional methods try not to settle project scope in the early phases of the project; rather they introduce greater flexibility to coping with user requirements. This paper covers new and interesting approaches for determining project scope using agile software development methods.